Problem Wk.5.2.1: Wall Finder System Function

Use `sf.R`, `sf.Gain`, `sf.Cascade`, `sf.FeedbackAdd` and `sf.FeedbackSubtract` to construct the wall-finder robot system function. These procedures are documented in `Module sf` in the software documentation.

Define a Python procedure called `controllerSF` that takes one argument:

- the proportional gain, `k`, applied to the error.

and which returns the system function of a system whose input is the distance error' and whose output is the commanded velocity.'

Define a Python procedure called `plantSF` that takes one argument:

- the time step duration `T`.

and which returns a system function for a system whose input is the commanded' velocity and whose output is the actual distance to the wall.'

Define a Python procedure called `sensorSF` that takes no arguments and which returns a system function for a system whose input is the actual sensor value and whose output is a one-step delayed sensor reading.

Define a Python procedure called `wallFinderSystemSF` that takes two arguments:

- the time step duration `T`.
- the proportional gain, `k`, applied to the error.

and which returns a system function for a system whose input is the desired distance' and whose output is the actual distance to the wall.'

You can debug these in Idle by placing your definitions in a file and including `import lib601.sf as sf` at the top.'

You can check if you re getting the right answer by comparing the system function that you get with the difference equations for the various systems that you derived in the problems for the previous design lab. You will need to convert your difference equations to system functions (or vice versa) to compare - beware of order of coefficients and minus signs...

```python
def controllerSF(k):
    pass

def plantSF(T):
    pass

def sensorSF():
    pass

def wallFinderSystemSF(T, k):
    pass
```

Problem Wk.5.3.2: Wall Follower System Function

Enter the system function of the proportional wall follower $H = \frac{D_o}{D_i}$ by entering the numerator and denominator polynomials in R. Enter the polynomials as expressions the way you would enter them in Python (although dropping * as in normal algebraic notation should also work). You can use the following symbols:

```
R, T, V, k
```

(these variables are case-sensitive)

An example of a polynomial in R entered in this format is:

```
 k*T*(R**2) - V*(T**2)*R + 1
```

- You need to enter both the numerator and denominator before you can check either. They will be checked together. **The check will be on whether the ratio is correct, not whether the individual parts are correct.**
- Make liberal use of parentheses to avoid ambiguity.

Numerator:

Denominator:

Problem Wk.5.3.4: Wall Follower Model

Use `sf.R`, `sf.Gain`, `sf.Cascade`, `sf.FeedbackAdd` and `sf.FeedbackSubtract` to construct the wall-finder robot system function. These procedures are documented in `Module sf` in the software documentation.

Define a Python procedure called `wallFollowerModel` that takes three arguments:

- the proportional gain, `k`, applied to the error.
- the time step duration `T`
- the translational velocity `V`

and which returns a system function for a system whose input is the desired distance and whose output is the actual distance to the wall.

You can debug these in Idle in the `designLab05Work.py` file.